PATENT ABSTRACTS OF JAPAN

(11)Publication number : 11-045220

(43)Date of publication of application : 16.02.1999

(51)Int.Cl. G06F 13/00
G06F 9/46

(21)Application number : 10-131557

(22)Date of filing : 14.05.1998

(71)Applicant : INTERNATL BUSINESS MACH CORP <IBM>

(72)Inventor : CHEN SHAWFU
ROBERT O DREYFUS
RAPYAN RAU
ROBERT B VANDONGEN
SHUFI WARNER
DANIEL L E

(30)Priority

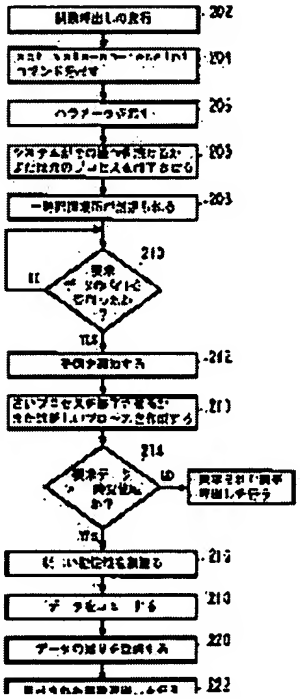
Priority number : 97 865166 Priority date : 29.05.1997 Priority country : US

(54) DEVICE AND METHOD FOR REDUCING SYSTEM RESOURCE BY USING TCP/IP SOCKET APPLICATION

(57)Abstract:

PROBLEM TO BE SOLVED: To introduce a data processing tool into a computer system which has at least one resident operating system.

SOLUTION: When a read/receive command is received, a special interface function call is issued and information regarding a program name and a token relating to read/reception request side application or a processor is held. Then a socket descriptor discriminating a data communication protocol is generated. Further, a system control table entry including the program name, token information, and information relating to the socket descriptor is generated. After data arrive, this file makes it possible to process the read/receive command and



send relative process information back to the request side application or processor.

LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

(19)日本国特許庁（ＪＰ）　(12) **公開特許公報（A）**　(11)特許出願公開番号

特開平11－45220

(43)公開日　平成11年(1999)２月16日

(51)Int.Cl.[6]	識別記号	ＦＩ	
Ｇ０６Ｆ　13/00	３５３	Ｇ０６Ｆ　13/00	３５３Ｎ
9/46	３６０	9/46	３６０Ｆ

審査請求　未請求　請求項の数26　ＯＬ　（全 10 頁）

(21)出願番号　特願平10－131557

(22)出願日　平成10年(1998)５月14日

(31)優先権主張番号　０８／８６５１６６

(32)優先日　1997年５月29日

(33)優先権主張国　米国（ＵＳ）

(71)出願人　390009531
インターナショナル・ビジネス・マシーンズ・コーポレイション
ＩＮＴＥＲＮＡＴＩＯＮＡＬ　ＢＵＳＩＮＥＳＳ　ＭＡＳＣＨＩＮＥＳ　ＣＯＲＰＯＲＡＴＩＯＮ
アメリカ合衆国10504、ニューヨーク州　アーモンク　（番地なし）

(72)発明者　シャウフ・チェン
アメリカ合衆国06776　コネチカット州ニュー・ミルフォード　メイプルウッド・ドライブ　８

(74)代理人　弁理士　坂口　博　（外１名）

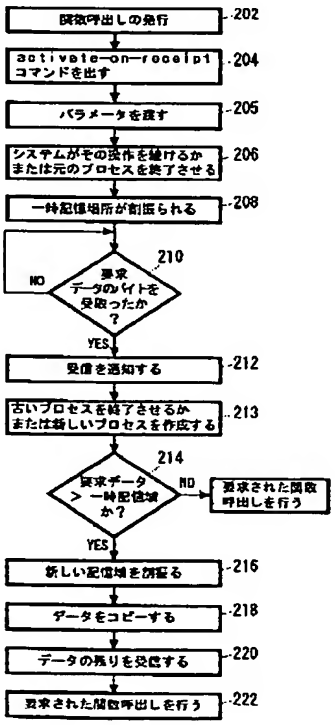
最終頁に続く

(54)【発明の名称】　ＴＣＰ／ＩＰソケット・アプリケーションを使用したシステム資源削減装置および方法

(57)【要約】

【課題】　少なくとも１つの常駐オペレーティング・システムを有するコンピュータ・システム内に、データ処理ツールを導入する。

【解決手段】　読取り／受信コマンドを受信したときに、特殊なインタフェース関数呼出しを発行し、読取り／受信要求側アプリケーションまたはプロセッサに関連するプログラム名およびトークンに関する情報を保存する。次いで、データ通信プロトコルを識別するソケット記述子を生成する。さらにまた、プログラム名、およびトークン情報およびソケット記述子に関する情報を含むシステム制御テーブル・エントリを生成する。データが到着した後、このファイルは、読取り／受信コマンドの処理および要求側アプリケーション／プロセッサへの関連する処理情報の返送を可能にする。



【特許請求の範囲】

【請求項１】少なくとも１つのオペレーティング・システムを有するコンピュータ・システム内で、

読取りコマンドを受信したときに特殊なインタフェース関数呼出しを発行するステップと、

前記読取りコマンドに関連するデータが着信データ用に取っておかれた一時記憶位置において受信されるまで前記コンピュータ・システムに他の動作を処理させるステップと、

前記オペレーティング・システムが前記読取りコマンドを発行したアプリケーションまたはプロセッサに関連するプログラム名およびトークンを保存するステップと、

データ通信プロトコルを識別するソケット記述子を生成するステップと、

前記プログラム名およびトークン情報をデータ構造フォーマット中に保存し、かつそれらを前記ソケット記述子に関連付けるファイル記述子を有するシステム制御テーブル・エントリを生成するステップと、

前記要求された読取りコマンドに関連するデータを受け取ったときに、前記システム制御テーブルを位置指定し、ソケット記述子情報がどこに常駐するかに関連する情報を識別することによって要求側アプリケーションまたはプロセッサを追跡するステップと、

前記コンピュータ・システム内で新しいプロセスを生成し、前記システム制御テーブルから得られた前記データを要求する前記プログラムの名前を入力するステップと、

前記読取りコマンドを処理するステップとを含むデータ処理方法。

【請求項２】前記コンピュータ・システムがそれ自体のバッファからか、またはそのオペレーティング・システムによって得られた他のバッファからデータを得ることができるように他の読取りコマンドを発行するステップをさらに含む、請求項１に記載の方法。

【請求項３】前記データが処理される前に前記一時記憶位置のサイズが前記着信データよりも小さい場合に他の特殊なプログラム・インタフェース関数呼出しを発行するステップをさらに含む、請求項１に記載の方法。

【請求項４】新しい記憶位置が前記着信データ用に割り振られ、かつ前記データが前記新しい記憶位置にコピーされる、請求項３に記載の方法。

【請求項５】ＴＣＰ／ＩＰ通信プロトロルが使用される、請求項１に記載の方法。

【請求項６】前記プログラム名が、前記プログラム名を指示するプログラム名ポインタと呼ばれるパラメータによって識別される、請求項１に記載の方法。

【請求項７】前記トークン名が、前記プログラム名を指示するトークン名ポインタと呼ばれるパラメータによって識別される、請求項１に記載の方法。

【請求項８】前記プログラム名ポインタが、前記プログラム名のアドレスを含み、前記データが到着したときに使用され、データ通信セッションを実施する間に他の点から受信される、請求項６に記載の方法。

【請求項９】前記トーク・ポインタが、前記アドレスが前記プロセス間に送っているデータを有する８バイト・トークンのアドレスを含む、請求項７に記載の方法。

【請求項１０】前記トーク名パラメータの長さが８バイトである、請求項９に記載の方法。

【請求項１１】前記ソケット記述子が前記動作中のアプリケーションの通信セッションに関連するパラメータである、請求項１に記載の方法。

【請求項１２】前記ソケット記述子が整数である、請求項１に記載の方法。

【請求項１３】少なくとも１つのオペレーティング・システムを有するコンピュータ・システム内で、

Ａｃｔｉｖａｔｅ＿ｏｎ＿ｒｅｃｅｉｐｔアプリケーション・プログラム・インタフェース関数呼出しを発行し、プロセスを終了するステップと、

前記読取りコマンドに関連するデータが受信されるまで前記コンピュータ・システムに他の動作を処理させるステップと、

前記オペレーティング・システムが前記Ａｃｔｉｖａｔｅ＿ｏｎ＿ｒｅｃｅｉｐｔコマンドを発行したアプリケーションに関連するプログラム名およびトークンを保存するステップと、

データ通信プロトコルを識別するソケット記述子を生成するステップと、

前記プログラム名およびトークン情報をデータ構造フォーマット中に保存し、かつそれらを前記ソケット記述子に関連付けるファイル記述子を有するシステム制御テーブルを生成するステップと、

前記要求された受信コマンドに関連するデータを受け取ったときに、前記システム制御テーブルを位置指定し、前記ソケット記述子情報が常駐すること発見するステップと、

前記コンピュータ・システム内で新しいプロセスを生成し、前記システム制御テーブルから得られた前記データを要求する前記プログラムの名前を入力するステップと、

前記読取りコマンドを処理するステップとを含むデータ処理方法。

【請求項１４】前記コンピュータ・システムがそれ自体のバッファからか、またはそのオペレーティング・システムによって得られた他のバッファからデータを得ることができるように他の受信コマンドを発行するステップをさらに含む、請求項１３に記載の方法。

【請求項１５】前記データが処理される前に前記一時記憶位置のサイズが前記着信データよりも小さい場合に他の特殊なプログラム・インタフェース関数呼出しを発行するステップをさらに含む、請求項１３に記載の方法。

【請求項１６】前記データが処理される前に前記一時記憶位置のサイズが前記着信データよりも小さい場合に他の特殊なプログラム・インタフェース関数呼出しを発行するステップをさらに含む、請求項１３に記載の方法。
【請求項１７】新しい記憶位置が前記着信データ用に割り振られ、かつ前記データが前記新しい記憶位置にコピーされる、請求項１６に記載の方法。
【請求項１８】前記受信コマンドがデータグラムまたはコネクションレス・ソケットに対して発行される、請求項１３に記載の方法。
【請求項１９】前記プログラム名が、前記プログラム名を指示するプログラム名ポインタと呼ばれるパラメータによって識別される、請求項１３に記載の方法。
【請求項２０】前記トークン名が、前記プログラム名を指示するトークン名ポインタと呼ばれるパラメータによって識別される、請求項１３に記載の方法。
【請求項２１】前記プログラム名ポインタが、前記プログラム名のアドレスを含み、前記データが到着したときに使用され、データ通信セッションを実施する間に他の点から受信される、請求項１９に記載の方法。
【請求項２２】前記トーク・ポインタが、前記アドレスが前記プロセス間に送っているデータを有する８バイト・トークンのアドレスを含む、請求項２０に記載の方法。
【請求項２３】前記トーク名パラメータの長さが８バイトである、請求項２２に記載の方法。
【請求項２４】前記ソケット記述子が前記動作中のアプリケーションの通信セッションに関連するパラメータである、請求項１３に記載の方法。
【請求項２５】前記ソケット記述子が整数である、請求項１３に記載の方法。
【請求項２６】前記受信コマンドがｒｅｃｖｆｒｏｍ（）関数呼出しである、請求項１３に記載の方法。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】本発明は、ＴＣＰ／ＩＰソケット・アプリケーション中などの通信セッション中に発行された読取り機能などの特定のコマンドを処理している間にシステムによって使用される資源を削減するように設計されたツールに関する。
【０００２】
【従来の技術】ホスト・システム・ネットワークまたはその他の同様のマルチホスト・コンピュータ環境では、いくつかのプロセッサとホストが共存し、それらが同時にデータの要求とアクセスを行ったり異なる常駐アプリケーションを実行したりする。そのような環境では、データ処理システムを使用してその環境全体の様々な場所で必要なデータを維持し、様々なアプリケーションの処理を行う。これによって、遠隔ホストまたはユーザに単一のイメージを提示することができ、プロセッサ複合体による負荷分散が可能になる。データは環境内の通信リンクのネットワークを形成する遠隔ホストおよびその他のホスト・システムにリンクされた１つまたは複数のホスト・システムで維持することができる。リンク上の１つのホストからリンク上の他のホストにメッセージを送信するために、プロトコルと呼ばれる通信規則を確立して、メッセージを経路指定し、複合体または環境内のリンク上の適切なホスト・コンピュータにアクセスすることによって通信を制御する。このような通信プロトコルは一般に、データ通信製品の機能と構造を規定する遠隔処理アーキテクチャの一部として存在する。
【０００３】このようなコンピュータ・ネットワークまたは環境では、１つまたは複数のオペレーティング・システムによって供給される機能インターフェースであるアプリケーション・プログラム・インタフェース（以下ＡＰＩと呼ぶ）も設けられ、高水準言語で書かれたアプリケーション・プログラムがオペレーティング・システムの特定のデータまたは機能を使用することができるようにする。場合によっては、ＡＰＩはアプリケーション・プログラムがそれを介してアクセス方法と対話するインタフェースとして機能する。マルチタスク・オペレーティング・システムでは、アプリケーション・プログラムの要求がＡＰＩを介してオペレーティング・システムに対して行われ、要求によって、実行するタスクまたはプロセスが自動的に開始される。並列処理環境では、コンピュータ・アーキテクチャは大量のタスクを高速で同時に処理するために、相互接続された多くのプロセッサを使用して大量のデータにアクセスする。このような環境では、マルチタスク・オペレーティング・システムは適時のタスク処理のためにＡＰＩに大きく依存している。
【０００４】必要なタスク処理を遅滞なく行うために、多くのＡＰＩはプロセスのグループ間で複合通信を行うための集合操作のセットを定義する。集合的操作には使いやすさやパフォーマンスを始め、それに付随するいくつかの利点がある。しかし、集合操作の使用には１つの大きな欠点がある。ほとんどの場合、それらの集合操作の多くは同期し、タスクを実行する時がくるまでプロセッサを「ブロック」する。このため「ブロッキング」操作と呼ばれる。そのタスクが本質的に同期されないアプリケーションの解決策は、「非ブロッキング」または非同期集合操作を使用することである。非ブロッキング操作によって、各タスクはそのタスク独自のペースで進むことができ、必要であれば完了または「待ち」状態になっていないか定期的に検査することもできる。しかし、非ブロッキング操作にはある種の欠点もある。たとえば、待ちが必要な場合、または内部依存関係に遭遇した場合、意志決定のためにユーザに制御を返す必要がある。これは、非同期集合操作を使用する非ブロッキング環境では、操作全体がタスクのいずれかまたはプロセッ

サ時間を「ブロック」することができないためである。この依存関係のために、プロセスのただの１段階も操作をブロックすることができない。

【０００５】しかし、ユーザに制御を返すことは問題の一部にすぎない。ユーザに制御が返された後の１つの大きな問題は、元の内部依存関係が解決された場合などに、いつ操作を再開することができるかを通知する問題である。第２の関連する問題は、ユーザが操作の再開を決定した後で、中断が行われる前と同じ処理箇所に戻ることである。

【０００６】ブロッキング操作と非ブロッキング操作の両方で発生する他の関連する共通の問題は、パフォーマンス問題とシステム遊休時間の解決である。ｒｅａｄ（）やｒｅｃｖｆｒｏｍ（）呼出などの特定のコマンドは、読取りコマンドまたは受信コマンドのデータが要求側アプリケーションによって受け取られるまでシステム環境が一時的に操作の処理を中断する必要がある。データを処理のためにただちに入手することができない場合、アプリケーションにとって使用可能な資源の大部分が中断状態になり、データが入手可能になるまではそれ以上処理が使用可能にならない。ブロッキング・モードでは、コマンド処理の順次性によってプロセス環境全体が強制的に着信データを待ってからでなければ、コマンド処理の次の段階を行うことができない。非ブロッキング・モードでは、やはりデータをただちに入手することができない場合、プロセス環境はデータの入手可能性を調べ続けなければならず、したがって非同期操作を同期操作に変えることになるか、または操作を終了して制御をユーザに返さなければならず、それによってさらに問題が生じる。高パフォーマンスのトランザクション指向システムでは、このようにデータを入手することができないことによって、多くのプロセスが一度にすべて中断する可能性がある。したがって、システムはどの新しいプロセスもディスパッチすることができないまま資源を使い果たすことがある。

【０００７】

【発明が解決しようとする課題および課題を解決するための手段】少なくとも１つの常駐オペレーティング・システムを有するコンピュータ・システム内に、データ処理ツールを導入する。ツールは、まず読取り／受信コマンドを受信したときに特殊なインタフェース関数呼出しを発行する。次いで、システムは、オペレーティング・システムが特定の読取り／受信コマンドを発行したアプリケーションまたはプロセッサに関連するプログラム名およびトークンを保存する間、他の動作を復帰または継続する。次いで、データ通信プロトコルを識別するソケット記述子を生成する。さらにまた、プログラム名およびトークン情報をデータ構造フォーマット中に保存し、かつそれらを前記ソケット記述子に関連付けるファイル記述子を有するシステム制御テーブル・エントリを生成する。データが到着した後、このシステム制御テーブルを見つけ出し、ソケット記述子の所在に関する記憶された情報を見つけることによって要求側アプリケーションまたはプロセッサを識別する。次いで、要求側プログラムの名前を得た後で新たに到着したデータを処理する新しいプロセスを生成し、最後に読取り／受信コマンドを処理する。

【０００８】

【発明の実施の形態】コンピュータ・ネットワークにいくつかのホストや遠隔ホストを接続する場合、それらの間に通信をセットアップし、維持する必要が生じる。通信を可能にするために、ホスト・システムのネットワークは通信リンクを含み、それらのリンクに異なる種類のホスト・コンピュータが接続される。リンク上の１つのホストからリンク上の他のホストにメッセージを送信するために、プロトコルと呼ぶ規則を確立して通信リンクを制御し、メッセージを経路指定し、リンク上の適切なホスト・コンピュータにアクセスする。

【０００９】通信プロトコルは概念的には図１に図示するような層状になっているものと見なすことができ、各プロトコル層はその下にある層によって提供されるサービスを利用する。最下層はリンク層１０２で、ハードウェア・レベルを扱い、特定のタイプの単一のネットワーク上のホスト間のデータの伝送を制御する。その上の層は、マシン対マシン（MM）層１０４で、同じ物理複合体に直接接続されていないホスト間で通信する機能を提供する。この層の広く使用されている例はインターネット・プロトコル（ＩＰ）である。インターネットＩＰは、標準ソフトウェア通信パッケージの使用を可能にする標準業界通信プロトコルである。

【００１０】プロトコルのその上の層はポート対ポート（ＰＰ）層１０６で、異なるアプリケーション・プログラムを実行する複数のプロセスが遠隔ホストにある遠隔プロセスと同時に通信することができるようにする。ＰＰ層はMMプロトコル層を使用してホスト機間でデータを転送する。ＰＰ層はアプリケーション層とインタフェースし、アプリケーション層はプロセスにローカル通信ポートを割り振り、そのポートを遠隔ホスト上の遠隔ポートに接続し、ローカル・ポートと遠隔ポートとの間でデータを転送する。このようなＰＰトランスポート・プロトコルの例としては、ＴＣＰ（転送制御プロトコル）、ＵＤＰ（ユーザ・データグラム・プロトコル）、およびＸＮＳ（ゼロックス・ネットワーク・システム）がある。ＴＣＰはＩＰプロトコル・スイート（ＴＣＰ／ＩＰ）を使用する装置と共に稼働することができる。

【００１１】最上層はアプリケーション層１０８である。ＡＰＩもこの層にある。さらに、ソケット・アプリケーションなどのその他の通信アプリケーションもこの層にある。ソケット・アプリケーションは、ポート識別子とＴＣＰ／ＩＰまたはその他の通信セッション・アド

レスとの連結によって作成された固有ホスト識別子である。

【００１２】いずれかの高水準アプリケーション、特にソケット・アプリケーションを稼働させている間、特定のタイプのコマンドによって処理の破壊や資源の一時的保留状態が生じることがある。これは、通常、データをただちに入手することができない場合である。たとえば、ＴＣＰプロトコルはデータにアクセスするためにｒｅａｄ（）コマンドを使用していた。ＵＤＰプロトコルは同様のコマンドｒｅｃｖ（）／ｒｅｃｖｆｒｏｍ（）関数呼出しを使用する。これはＴＣＰプロトコルのｒｅａｄ（）コマンドと非常によく似ている。どちらの場合も、ｒｅａｄ（）、ｒｅｃｖ（）、またはｒｅｃｖｆｒｏｍ（）コマンドでは、着信データを見越してメモリのかなりの部分がただちに割り振られ、プロセッサによって確保される。メモリのうちのこの割り振られたブロックは通常、最大長または最大データ長と呼ばれ、３２Ｋないし２メガの範囲とすることができる。

【００１３】データがただちに受信されない場合、アプリケーションにとって使用可能な資源の大部分が中断状態になり、データが入手可能になるまでそれ以上処理を行うことができない。高パフォーマンス・システムでは、これによって同時に多くのプロセスが中断する可能性がある。このシステムまたはプロセッサの使用不能状態は、以下で詳述するようにシステムがブロッキング集合操作と非ブロッキング集合操作のどちらを稼働させているかに関係なく生じる。

【００１４】Ａ）ブロッキング操作

アプリケーションまたはシステムがブロッキング・モードで稼働している場合、処理操作は同期モードで行われる。ｒｅａｄ（）、ｒｅｃｖ（）、またはｒｅｃｖｆｒｏｍ（）コマンドが発行されたがデータが直ちに使用可能にならず、システム全体が同期して稼働している場合、すべてのスレッドはそのデータ・バイトが最終的に着信する時点まで遊休状態になっていなければならない。データが入手可能になる時点までその先の処理を行うことができない。さらに、メモリの大きな部分をそのデータの受信用に割り振らなければならないため、システム環境全体が影響を受けることがある。

【００１５】Ｂ）非ブロッキング操作

非ブロッキング操作を稼働させている間は、システムの処理は非同期であり、したがってデータ読取り／受信コマンドが発行された後、プロセスはブロッキング操作の場合と同じ意味では遊休状態にならない。それにもかかわらず、データを受け取る前のプロセスの操作全体はブロッキング・モードで稼働しているときときわめてよく似ている。非ブロッキング・モードではシステムが読み取るデータの可用性を調べ、データがまだ使用不能であるとわかるとその都度、決定を行わなければならない。この決定は、データの可用性を絶えず検査し続け、それによってブロッキング操作とほとんど同じ状況を生じさせるか、あるいは、割り振られたメモリ空間を抹消し、現在遊休状態になっているプロセッサの連続動作に必要な空間割振りを作成し直すことによって、再割り当てする必要があるかのいずれかである。

【００１６】ブロッキングと非ブロッキングのいずれの場合も、最大長（最大データ長）すなわちデータの受信のために割り振られたメモリによって高パフォーマンス・トランザクション・プロセス・システム内で使用可能な資源の途切れのない可用性に大きな問題が生じる。

【００１７】本発明は、ａｃｔｉｖａｔｅ＿ｏｎ＿ｒｅｃｅｉｐｔ機構によって上述の問題を解決する。図２に示すように、（２０２）でｒｅａｄ（）、ｒｅｃｖ（）、ｒｅｃｖｆｒｏｍ（）またはその他の同様の関数呼出しまたはコマンドを受信すると、本発明はａｃｔｉｖａｔｅ＿ｏｎ＿ｒｅｃｅｉｐｔ（）コマンドと呼ぶ新しい関数呼出し（ＡＰＩ）を出す（２０４）。基本的には、この新しい関数呼出しによってプロセスの操作があたかもコマンドがまったく出されなかったかのように続けられ（２０６）、それによって元のプロセスはデータの最初のバイトが着信する時点まで稼働する。しかし、状況によっては、読取り／受信が発行された後でプロセスを実際に終了させる必要があることもある。本発明の代替実施形態では、この問題を解決する。そのような場合、読取り／受信の最初の発行によって開始されたプロセスをまず終了させて資源が解放されるようにする。しかし、ａｃｔｉｖａｔｅ＿ｏｎ＿ｒｅｃｅｉｖｅコマンド呼出しは、そのような終了にもかかわらず要求データの入手を続け、データが着信すると、前述の場合のように新しいプロセスが作成される。いずれの場合も、着信データの少なくとも一部を受信するようにメモリ内に好ましくは４Ｋの一時記憶場所を割り振る（２０８）。これによって、最大２メガもの大きな部分を使用する従来技術とは異なり、そのようなデータの受信のためにメモリの小領域が確保される。さらに、最初の４Ｋのメモリはプロセッサではなくシステム自体によってそのつど割り振られ、それによって空間とメモリの可用性が最も効率的に使用される。これは、システムには、プロセッサのうちのいずれか単独のプロセッサよりもそのような可用性がよく見えるため、システムは使用可能なポケットがあればどのポケットにでも記憶域を割り振ることができるからである。

【００１８】各ａｃｔｉｖａｔｅ＿ｏｎ＿ｒｅｃｅｉｐｔ呼出しと共にいくつかのパラメータも渡され（２０５）、後の処理を可能にする。図３に示すように、渡されるパラメータは、そのコマンドまたは呼出しを出したアプリケーションおよびホストを後で識別するためのソケット記述子、プログラム名ポインタ、およびトークン・ポインタである（３０３）。ソケット記述子は、前のｓｏｃｋｅｔ（）ＡＰＩ関数呼出しからアプリケーショ

ンが入手した整数であり、ＴＣＰ／ＩＰまたはその他の同様のプロトコルのデータ通信セッションのうちのアプリケーション側に関連づけられる（３３２）。プログラム名ポインタには、プロトコルのデータ通信の他方の側からホストにデータが着信したときに入れられるプログラム名のアドレスが入る。トークン・ポインタには、アプリケーションがプロセス間で受け渡しするデータを含む８バイトのトークンのアドレスが入れられる。

【００１９】ソケット・アプリケーションがａｃｔｉｖａｔｅ＿ｏｎ＿ｒｅｃｅｉｐｔ（）関数呼出しを出した後、オペレーティング・システムはプログラム名と８バイト・トークンを、そのソケット記述子に関連づけられたファイル記述子と呼ばれるデータ構造に保管する（３３４）。データの着信が開始された後、オペレーティング・システムはａｃｔｉｖａｔｅ＿ｏｎ＿ｒｅｃｅｉｐｔ（）をｒｅａｄ（）またはｒｅｃｖｆｒｏｍ（）関数呼出しに変換する。ストリームまたはコネクション指向（ＴＣＰ）ソケットに対してａｃｔｉｖａｔｅ＿ｏｎ＿ｒｅｃｅｉｐｔ（）が発行された場合、システムはそのａｃｔｉｖａｔｅ＿ｏｎ＿ｒｅｃｅｉｐｔ（）関数呼出しをｒｅａｄ（）関数呼出しに変換する。データグラムまたはコネクションレス（ＵＤＰ）ソケットに対してａｃｔｉｖａｔｅ＿ｏｎ＿ｒｅｃｅｉｐｔ（）関数呼出しが出された場合、システムはそのａｃｔｉｖａｔｅ＿ｏｎ＿ｒｅｃｅｉｐｔ（）関数呼出しをｒｅｃｖｆｒｏｍ（）関数呼出しに変換する。ｒｅａｄ（）またはｒｅｃｖｆｒｏｍ（）は、最大長（３２Ｋまたはそれ以上の要求）のＴＣＰ／ＩＰスタックに対して発行される。

【００２０】図２（２１４）に示すように、データの最初のバイトが着信するとこの機構に通知される。次にこの機構は、割り振られた一時記憶場所が着信データを収容することができる大きさであるかどうかを判断する。

【００２１】受信したデータが４Ｋの割振り空間より小さいか等しい場合、受信するデータの残りを受信するためにそれ以上の空間を割り当てない。しかし、データが４Ｋより長い場合、受信するデータ全体を収容するためにさらに空間が割り振られる（２１６）。

【００２２】割り振られた場所に最初のデータ・バイトを受信すると、ａｃｔｉｖａｔｅ＿ｏｎ＿ｒｅｃｅｉｐｔ（）コマンドは、元のプロセスを終了させることでシステムを「活動化」するかまたはホストで新しいプロセスを作成する。次にデータの読取り、処理、または転送を行う（２１３）。（データが着信した後、システムは新しい記憶域を入手してそのデータをその新しい記憶域にコピーする必要がある場合がある（２１８）。後で、データを受信すると関数呼出しを行うことができる（２２０～２２２）。）このプロセスは詳細には以下のように行われる。

【００２３】図３に示すように、ファイル記述子を使用して、システム制御テーブルで、前に保管していたプログラム名および８バイト・トークンと、当該ＴＣＰ／ＩＰソケットに関連づけられたソケット記述子を探し出す（３３８）。システムはそのソケット記述子、８バイト・トークン、記憶ブロックのアドレス、および内部ヘッダから入手したデータの長さを、システムによって作成される新しいプロセスに関連づけられた別の制御ブロックにコピーする。ファイル記述子によって、メッセージがデータグラム・ソケットから受信されたことが示されている場合、システムはメッセージのソース・アドレスを内部ヘッダから新しいプロセスに関連づけられたブロックにコピーする。システムは次に新しいプロセスを作成し（３３９）、ファイル記述子から名前を入手したプログラムに入る。

【００２４】データを入手するために、新しいプログラムは、プロセスでシステムから渡されたデータを使用して標準のｒｅａｄ（）、ｒｅｃｖ（）、またはｒｅｃｖｆｒｏｍ（）ソケットＡＰＩ関数呼出しを出す。システムはシステムがすでにネットワークからデータを入手していることを知っているため、この関数呼出しはＴＣＰ／ＩＰまたはその他の同様のプロトコルのスタックには送られない。この第２のｒｅａｄ（）、ｒｅｃｖ（）、またはｒｅｃｖｆｒｏｍ（）関数呼出しの目的は、プログラムがそれ自体のバッファから、またはオペレーティング・システムから入手したバッファから、データを入手することができるようにすることである。アプリケーションがｒｅａｄ（）、ｒｅｃｖ（）、またはｒｅｃｖｆｒｏｍ（）関数呼出しを出した後、システムはアプリケーションに対してアプリケーションによって指定されたバッファにデータを返す。

【００２５】本発明の好ましい実施形態では、割振り空間でデータの最初のバイトを受信した後でデータが４Ｋの割振り空間よりも大きい場合、（３４５）に示すようにデータを入手するためにｒｅａｄ（）、ｒｅｃｖ（）、またはｒｅｃｖｆｒｏｍ（）コマンドを出し、それによって着信データの転送またはコピーあるいはその両方のために最大長またはその他の適切な長さの空間を割り振る。しかし、着信データが最初に割り振った４Ｋの空間に入る場合は、上記の場合のｒｅａｄ（）、ｒｅｃｖ（）、またはｒｅｃｖｆｒｏｍ（）コマンドの代わりにもう一度ａｃｔｉｖａｔｅ＿ｏｎ＿ｒｅｃｅｉｐｔコマンドを出す（３５０）。この第２のａｃｔｉｖａｔｅ＿ｏｎ＿ｒｅｃｅｉｐｔ関数呼出しによって、それ以上メモリ空間を割り振る必要なしに、割り振られた４Ｋのメモリでシステムによるデータの処理が可能になる。言い換えると、その場合、より長いデータ・バイトのコピー操作に対応するために最大長空間を割り振らない。

【００２６】まとめとして、本発明の構成に関して以下の事項を開示する。

【００２７】（１）少なくとも１つのオペレーティング

・システムを有するコンピュータ・システム内で、読取りコマンドを受信したときに特殊なインタフェース関数呼出しを発行するステップと、前記読取りコマンドに関連するデータが着信データ用に取っておかれた一時記憶位置において受信されるまで前記コンピュータ・システムに他の動作を処理させるステップと、前記オペレーティング・システムが前記読取りコマンドを発行したアプリケーションまたはプロセッサに関連するプログラム名およびトークンを保存するステップと、データ通信プロトコルを識別するソケット記述子を生成するステップと、前記プログラム名およびトークン情報をデータ構造フォーマット中に保存し、かつそれらを前記ソケット記述子に関連付けるファイル記述子を有するシステム制御テーブル・エントリを生成するステップと、前記要求された読取りコマンドに関連するデータを受け取ったときに、前記システム制御テーブルを位置指定し、ソケット記述子情報がどこに常駐するかに関連する情報を識別することによって要求側アプリケーションまたはプロセッサを追跡するステップと、前記コンピュータ・システム内で新しいプロセスを生成し、前記システム制御テーブルから得られた前記データを要求する前記プログラムの名前を入力するステップと、前記読取りコマンドを処理するステップとを含むデータ処理方法。

（2）前記コンピュータ・システムがそれ自体のバッファからか、またはそのオペレーティング・システムによって得られた他のバッファからデータを得ることができるように他の読取りコマンドを発行するステップをさらに含む、上記（1）に記載の方法。

（3）前記データが処理される前に前記一時記憶位置のサイズが前記着信データよりも小さい場合に他の特殊なプログラム・インタフェース関数呼出しを発行するステップをさらに含む、上記（1）に記載の方法。

（4）新しい記憶位置が前記着信データ用に割り振られ、かつ前記データが前記新しい記憶位置にコピーされる、上記（3）に記載の方法。

（5）ＴＣＰ／ＩＰ通信プロトロルが使用される、上記（1）に記載の方法。

（6）前記プログラム名が、前記プログラム名を指示するプログラム名ポインタと呼ばれるパラメータによって識別される、上記（1）に記載の方法。

（7）前記トークン名が、前記プログラム名を指示するトークン名ポインタと呼ばれるパラメータによって識別される、上記（1）に記載の方法。

（8）前記プログラム名ポインタが、前記プログラム名のアドレスを含み、前記データが到着したときに使用され、データ通信セッションを実施する間に他の点から受信される、上記（6）に記載の方法。

（9）前記トーク・ポインタが、前記アドレスが前記プロセス間に送っているデータを有する８バイト・トークンのアドレスを含む、上記（7）に記載の方法。

（１０）前記トーク名パラメータの長さが８バイトである、上記（9）に記載の方法。

（１１）前記ソケット記述子が前記動作中のアプリケーションの通信セッションに関連するパラメータである、上記（1）に記載の方法。

（１２）前記ソケット記述子が整数である、上記（1）に記載の方法。

（１３）少なくとも１つのオペレーティング・システムを有するコンピュータ・システム内で、Ａｃｔｉｖａｔｅ＿ｏｎ＿ｒｅｃｅｉｐｔアプリケーション・プログラム・インタフェース関数呼出しを発行し、プロセスを終了するステップと、前記読取りコマンドに関連するデータが受信されるまで前記コンピュータ・システムに他の動作を処理させるステップと、前記オペレーティング・システムが前記Ａｃｔｉｖａｔｅ＿ｏｎ＿ｒｅｃｅｉｐｔコマンドを発行したアプリケーションに関連するプログラム名およびトークンを保存するステップと、データ通信プロトコルを識別するソケット記述子を生成するステップと、前記プログラム名およびトークン情報をデータ構造フォーマット中に保存し、かつそれらを前記ソケット記述子に関連付けるファイル記述子を有するシステム制御テーブルを生成するステップと、前記要求された受信コマンドに関連するデータを受け取ったときに、前記システム制御テーブルを位置指定し、前記ソケット記述子情報が常駐すること発見するステップと、前記コンピュータ・システム内で新しいプロセスを生成し、前記システム制御テーブルから得られた前記データを要求する前記プログラムの名前を入力するステップと、前記読取りコマンドを処理するステップとを含むデータ処理方法。

（１４）前記コンピュータ・システムがそれ自体のバッファからか、またはそのオペレーティング・システムによって得られた他のバッファからデータを得ることができるように他の受信コマンドを発行するステップをさらに含む、上記（１３）に記載の方法。

（１５）前記データが処理される前に前記一時記憶位置のサイズが前記着信データよりも小さい場合に他の特殊なプログラム・インタフェース関数呼出しを発行するステップをさらに含む、上記（１３）に記載の方法。

（１６）前記データが処理される前に前記一時記憶位置のサイズが前記着信データよりも小さい場合に他の特殊なプログラム・インタフェース関数呼出しを発行するステップをさらに含む、上記（１３）に記載の方法。

（１７）新しい記憶位置が前記着信データ用に割り振られ、かつ前記データが前記新しい記憶位置にコピーされる、上記（１６）に記載の方法。

（１８）前記受信コマンドがデータグラムまたはコネクションレス・ソケットに対して発行される、上記（１３）に記載の方法。

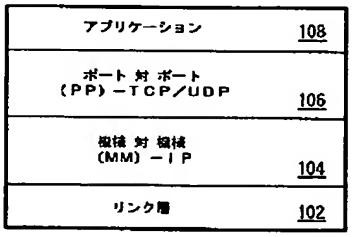
（１９）前記プログラム名が、前記プログラム名を指示

するプログラム名ポインタと呼ばれるパラメータによって識別される、上記（１３）に記載の方法。
（２０）前記トークン名が、前記プログラム名を指示するトークン名ポインタと呼ばれるパラメータによって識別される、上記（１３）に記載の方法。
（２１）前記プログラム名ポインタが、前記プログラム名のアドレスを含み、前記データが到着したときに使用され、データ通信セッションを実施する間に他の点から受信される、上記（１９）に記載の方法。
（２２）前記トーク・ポインタが、前記アドレスが前記プロセス間に送っているデータを有する８バイト・トークンのアドレスを含む、上記（２０）に記載の方法。
（２３）前記トーク名パラメータの長さが８バイトである、上記（２２）に記載の方法。
（２４）前記ソケット記述子が前記動作中のアプリケーションの通信セッションに関連するパラメータである、上記（１３）に記載の方法。
（２５）前記ソケット記述子が整数である、上記（１３）に記載の方法。（２６）前記受信コマンドがｒｅｃｖｆｒｏｍ（）関数呼出しである、上記（１３）に記載の方法。

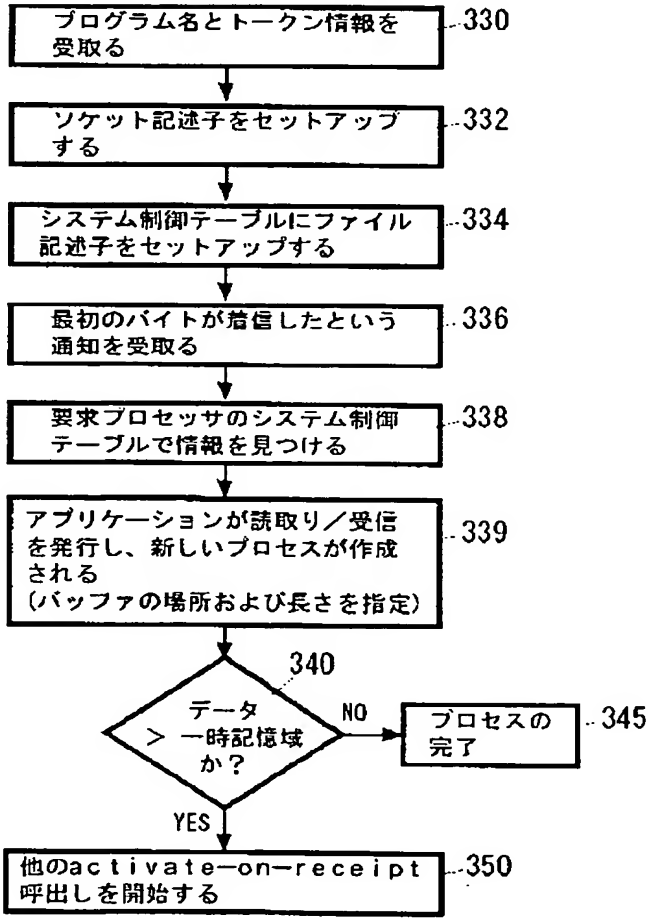
【図面の簡単な説明】
【図１】使用する通信プロトコルの概念層を示すブロック図である。
【図２】本発明の概要を示すフローチャートである。
【図３】本発明の特定の実施形態を示すフローチャートである。

【符号の説明】
１０２　リンク層
１０４　マシン対マシン（ＭＭ）層
１０６　ポート対ポート（ＰＰ）層
１０８　アプリケーション層

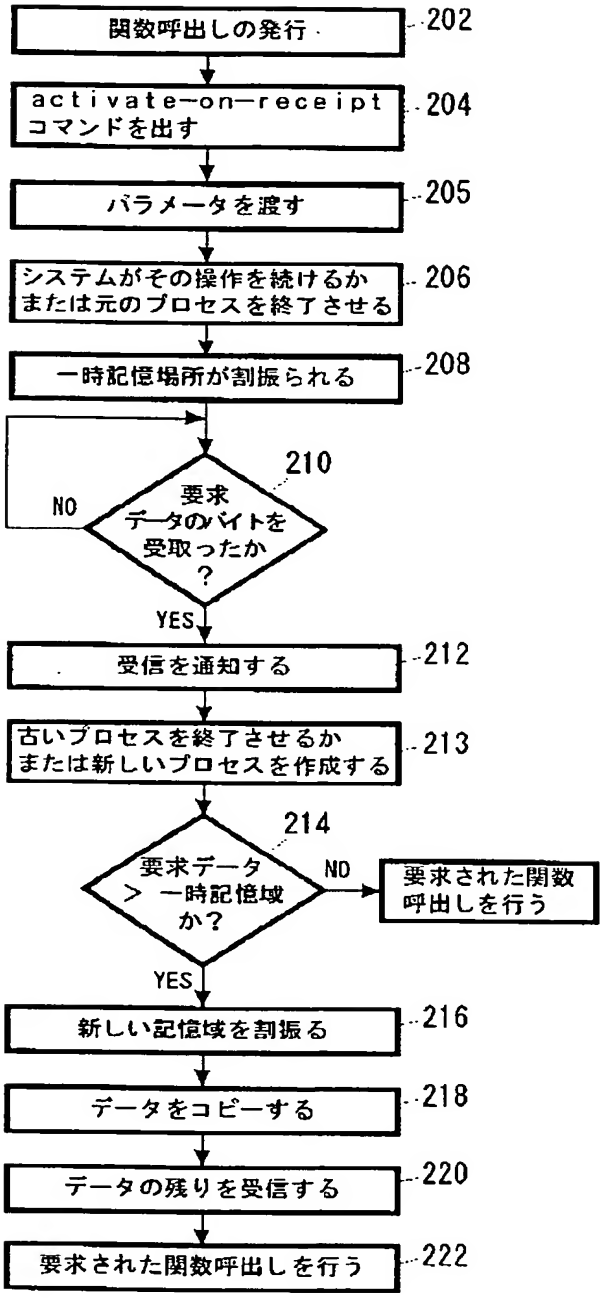
【図１】



【図３】



【図２】



フロントページの続き

(72)発明者　ロバート・オー・ドライフース
アメリカ合衆国12603　ニューヨーク州ポーキプシー　マンダレイ・ドライブ　14

(72)発明者　ラブヤン・ラウ
アメリカ合衆国12603－0704　ニューヨーク州ポーキプシー　スプリングサイド・ロード　7

(72)発明者 ロバート・ビー・ヴァンドンゲン
アメリカ合衆国12540 ニューヨーク州ラグランジュビル バーモア・ロード 54
(72)発明者 シュフイ・ワーナー
アメリカ合衆国06804 コネチカット州ブルックフィールド エドナ・コート 12
(72)発明者 ダニエル・エル・イー
アメリカ合衆国06801 コネチカット州ベセル シャロン・コート 11